Universidade do Porto

Faculdade de Engenharia

FEUP

Aplicação de ID3 ou C4.5 ao diagnóstico de doença renal crónica

Relatório Final

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do grupo:

Luis Oliveira - 201304515 - up201304515@fe.up.pt

Miguel Pereira - 201305998 - up201305998@fe.up.pt

Marta Lopes - 201208067 - ei12106@fe.up.pt

21 de Maio de 2016

Conteúdo

1 Objetivo

Este projeto, desenvolvido na unidade curricular de Inteligência Artificial, consiste na aplicação do algoritmo C5.0 para o diagnóstico da Doença Renal Crónica.

O programa deverá, através do algoritmo C5.0 e com base num conjunto de dados previamente disponibilizados, determinar uma Árvore de Decisão que traduz as regras de classifcação desse conjunto de dados. Estes dados vão ser então analisados de forma a verificar a eventual necessidade de pré-processamento. O modelo obtido deve poder depois ser utilizado na classificação de novos casos.

O objetivo final deste projeto será a determinação da árvore de decisão que traduz essas regras no diagnóstico da doença.

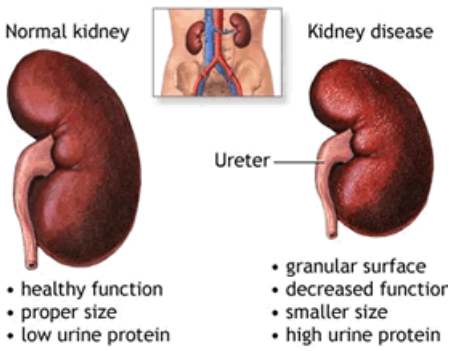
2 Especificação

2.1 Análise ao tema

2.1.1 Detalhes do tema

Neste projeto foi abordada a Doença Renal Crónica, esta doença signfica a presença de alterações da estrutura ou funções dos rins, com ou sem alteração da filtração glomerular, por um período maior que 3 meses e com implicações de saúde do indívudo.

Para identificar a lesão renal, são utilizados marcadores (ou seja, sintomas) que o paciente vai ter que vão permitir diagnosticar a doença: a presença de albuminúria maior que 30mg/24h, anormalidades no sedimento urinário, distúrbios eletrolíticos e outras desordens devido a doença dos túbulos renais, anormalidades detetadas por biópsia renal, exames de imagem, antecedentes de transplante renal e taxa de filtração glomerular menor que 60ml/min.



2.1.2 Ilustração de cenários

Esta ferramenta, criada por nós, poderá ser usada num cenário em que um médico necessitaria de apoio no diagnóstico da doença. O médico poderia usar esta aplicação para confirmar o diagnóstico da doença de um paciente, ou para esclarecer qualquer dúvida no diagnóstico final.

Achamos que também seria útil no auto-diagnóstico de qualquer pessoa que quisesse ter informações sobre o seu estado de saúde em relação à Doença Renal Crónica.

2.1.3 Training data set

Este conjunto de dados foi obtido no ano passado, após 2 meses de estudo nos Hospitais *Apollo*, na India, existem ao todo 25 atributos no *data set* dos quais 11 são numéricos e 14 nominais. Neste conjunto de dados vão existir 280 entradas, das quais 170 são indivíduos diagnosticados com a Doença Renal Crónica e 110 são saudáveis. Este conjunto de dados vai ser usado para a ferramenta conseguir determinar uma Árvore de Decisão que para diagnosticar a doença.

2.1.4 Test data set

O *test data set* foi obtido de maneira semelhante ao *training data set*, tem um total de 120 entradas dos quais 80 são indivíduos diagnosticados com a Doença Renal Crónica e 40 são saudáveis. Este conjunto de dados vai ser onde será decidido se o indívduo é saudável ou portador da doença, de acordo com as regras de decisão criadas a partir do *training data set*, validando os resultados obtidos.

2.2 Abordagem

2.2.1 Técnicas

Boosting

Esta técnica, incorporada no C5.0 e baseada no trabalho de Rob Schapire e Yoav Freund, vai gerar vários classificadores em vez de apenas um. Quando um caso novo precisa de ser classificado, cada classificador vota na classe onde prevê que esse caso pertence. Os votos são então contados na sua totalidade para determinar a classe final.

Primeiramente a árvore de decisão é gerada a partir dos casos de treino. O classificador, geralmente, vai classificar erradamente alguns dos casos de

treino. É então criado um segundo classificador que vai prestar mais atenção aos casos que foram classificados incorretamente, de forma a tentar que sejam classificados de forma correta pela nova árvore de decisão. O segundo classificador irá ser então diferente do primeiro, contudo, o segundo classificador também poderá cometer classificações erradas, e estas tornam-se foco de atenção durante a construção do terceiro classificador e assim sucessivamente até ser atingido um número pré determinado de iterações ou até o classificador mais recente ser extremamente preciso.

Winnowing

O algoritmo C5.0 consegue também prever quais atributos serão relevantes na classificação e quais não são, esta técnica é normalmente usada para ldiar com *datasets* de grandes dimensões.

Poda da Árvore de Decisão

Esta técnica consiste na remoção de partes de uma árvore de decisão, isto pode ocorrer porque não há uma contribuição significativa para a precisão ou intepretação da árvore, reduzindo-se a complexidade da árvore aumentado-se a sua generalização.

2.2.2 Algoritmos e sua breve explicação

Input: Exemplo, Atributo Alvo, Atributo.

Output: Árvore de Decisão.

Algoritmo:

1. Verificar classe base
2. Construir uma *decision tree* (DT) usando o *training data set*
3. Descobrir o atributo com **maior ganho de informação**
4. Para cada atributo ti que pertença a D, aplicar a DT para determinar a sua classe uma vez que a aplicação de um dado tuplo para uma DT é relativamente simples.

Casos base:

1. Todos os exemplos do *training data set* pertencem à mesma classe, isto é, uma folha da árvore marcada com essa classe é retornada;
2. O *training data set* estando vazio, é retornada uma folha caracterizada por insuficiência;
3. A lista de atributos, estando vazia, é devolvida uma folha contendo a classe mais frequente ou a disjunção de todas as classes.

2.3 Métricas

A grande dificuldade deste algortimo baseia-se em descobrir qual o algoritmo pelo qual se vai exercer a decisão. Este atributo pode ser o inicial ou um intermédio, e o método pelo qual se escolhe está presente na métrica que o algoritmo contempla.

A métrica utilizado pelo algoritmo C5.0 é o **ganho de informação** que consiste em obter o maior número de elementos dos conjuntos puros. Estes conjuntos são aqueles em que os resultados se direcionam todos para o mesmo lado, isto é, ou apontam todos para o **sim** ou apontam todos para o **não**. A **pureza** de um cojunto é obtida a partir da medição da **incerteza** de uma classe e esta é dada a partir do cálculo da *entropia* da sua divisão.

A fórmula de ganho será então:

$$Gain(S,\ A)\ =\ H(S)\ -\ \sum_{V\ \in Values(A)} \frac{|Sv|}{|S|} H(Sv)$$

V - Possíveis valores de **A**.
S - Conjunto de exemplos **{Y}**.
Sv - Subconjunto onde $\mathbf{Ya = V}$.
H(S) - Cálculo da entropia.

A entropia devolverá o número de *bits* necessários para dizer se Y é positivo ou negativo e é calculada pela seguinte fórmula:

$$H(S)\ =\ -\ p_+ * log_2 p_+\ -\ p_- * log_2 p_-$$

S - Subconjunto de exemplos de treino.
$\mathbf{p_+/p_-}$ - Percentagem de exemplos positivos/negativos em **S**.

2.4 Heurísticas

A heurística de seleção de teste original baseada em ganho de informação não era suficiente, era necessário resolver problemas como os atributos com um grande número de resultados serem favoráveis em relação aos atributos com um menor número e alguns atributos dividiam os dados num grande número de *singletons*, ou seja, classes únicas. Estas classes possuíam então um nível de pureza máximo, por serem únicas. Foi assim necessário criar algo que contrariasse estes problemas, foi então criado o **GainRatio** que é realizada posteriormente ao **Gain** e utiliza outra fórmula (**Split** para prever os problemas anteriormente mencionados.

$$Split(S,\ A)\ =\ -\ \sum_{V\ \in\ Values(A)} \frac{|Sv|}{|S|} log \frac{|Sv|}{|S|}$$

A - Atributo candidato
V - Possíveis valores de **A**.
S - Conjunto de exemplos **{Y}**.
Sv - Subconjunto onde $\mathbf{Ya = V}$.

$$GainRatio(S,\ A)\ =\ \frac{Gain(S,\ A)}{Split(S,\ A)}$$

Esta heurística é utilizada no algoritmo C5.0.

3 Desenvolvimento

3.1 Ferramentas e Ambientes de Desenvolvimento

O sistema operativo utilizado para desenvolver o projeto foi Linux Mint 17.1 - Cinnamon. A linguagem de programação utilizada foi C++, usando como IDE o Eclipse. Decidimos usar a ferramenta See5 para o *data mining*, que utiliza o algoritmo C5.0 e permite descobrir padrões que formam categorias e constroiem classificadores para prever os resultados sobre o dominio em análise.

3.2 Detalhes Revelantes

Decidimos então escolher o algoritmo C5.0, que tem como base o algoritmo C4.5 dado nas aulas, mas que tem algumas evoluções. Os *rulesets* no algoritmo C5.0 vão ter **taxas de erro** menores e o uso de **memória** vai ser bastante mais eficiente na construção dos *rulesets*. Os *rulesets* vão ser também menores no C5.0, tal como as **árvores de decisão** serão mais simples e coesas. A **velocidade** também será bastante maior no algoritmo C5.0 conseguindo acabar em 73 segundos uma tarefa que demoraria 9 horas a terminar no algoritmo C4.5.

Neste algoritmo vai ser ainda possível a utlização de **boosting** para construir várias árvores de decisão, tentando melhorar as anteriores, fazendo assim previsões mais precisas. No C5.0 é possível também ponderar diferentes atributos e tipos de classificações incorretas. A ferramenta permite ainda resolver a questão do *label noise*, que ocorre quando os valores dos atributos de duas instâncias são exatamente iguais, mas diferem do resultado final.

No nosso projeto foram utilizadas ainda várias *flags* que vão alterar a forma como vão ser obtidos os resultados:
-b : utiliza boosting
-w : utiliza winnowing
-g : não utiliza poda

Estes três métodos já foram explicados previamente na secção 2.2.1.

4 Experiências

4.1 Objetivos

4.1.1 Construção de classificadores

4.1.2 Avaliação

4.2 Resultados

5 Conclusão

Após a primeira fase de testes ao algoritmo e com a utilização da *framework* See5/C5.0, podemos concluir que é importante proceder a uma análise cuidada e pormenorizada dos atributos e verificar se todos serão necessários para a criação da árvore, para assim levar a uma maior taxa de acertos. É importante poder utilizar a *framework* sem dificuldades e para isso foram criadas as funções necessárias para um fácil carregamento de dados e processamento dos mesmos.

Contudo, podemos afirmar que todo o trabalho desenvolvido até agora contribuiu para o desenvolvimento dos nossos conhecimentos acerca de sistemas de aprendizagem simbólicos automáticos, mais especificamente a aplicação e funcionamento do algoritmo C4.5. Concluimos que este algoritmo tem bastantes potencialidades na ajuda da previsão e deteção da Doença Renal Crónica, e até talvez de outras doenças, de uma forma rápida e eficiente.

6 Recursos

6.1 Bibliografia

See5/C5.0: *https://www.rulequest.com/r210.html*
C5.0: An Informal Tutorial: *https://www.rulequest.com/see5-unix.html*
Wikipédia - Algoritmo C4.5: *https://pt.wikipedia.org/wiki/Algoritmo_C4.5*

6.2 Software

Linux Mint 17.1 - Cinnamon Eclipse

6.3 Recursos

TeXworks: *https://www.tug.org/texworks/*
Github: *https://www.github.com*